IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In the Matter of the Application of:	David M. Chess et al.	
Serial No.:	10/696,200	Confirmation No.: 7325
Filed: October 28, 2003		
For: System Method and Program Product for Detecting Malicious Software		
Examiner:	Daniel L. Hoang	Group Art Unit: 2136
Attorney Docket No.:	GB920030050US1	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Sir:

In response to the Office Action of December 22, 2008, and in support of the Notice of Appeal file herewith, Applicants respectfully submit this Appeal Brief.

(I) Real Party in Interest

The real party in interest for this Application is assignee INTERNATIONAL BUSINESS MACHINES CORPORATION of Armonk, NY.

(II). Related Appeals and Interferences

There are no related appeals or interferences.

(III). Status of Claims

Claims 1-15 are pending and stand finally rejected. Claims 1-15 are being appealed.

(IV). Status of Amendments

No response was filed to the final rejection of December 22, 2008 prior to the present appeal. No amendments to the claims have been filed after the claims were finally rejected.

(V). Summary of claimed subject matter

(V.A) Claim 1

Claims 1 is directed to a method (Figs. 2B, 6, 7A-C) for detecting malicious software within or attacking a computer system (page 1 lines 7-9, page 3 lines 13-14, page 4 lines 1-18, page 7 lines 3-10). The method comprises executing a hook routine (step 211 in Fig.2B, 600, 610, 620 Fig. 6, page 4 lines 2-10, page 7 lines 26-31, page 9 lines 19-20, page 11 lines 13-31, page 12 lines 1-30, page 13 lines 7- 30, page 14 lines 7-31, page 15, lines 1-25) in response to a system call (step 206in Fig 2B, steps 601, 611, 621 in Fig. 6, page 4 lines 5-6, page 9 lines 17-20 and 26-28, page 11 lines 13-15 and 27-

29 and 31, page 12 lines 1-3 and 5-8 and 10-12 and 14-17, page 14 lines 6-7 and 20-23, page 15 lines 1-5). The hook routine is located at the location identified by the system call (page 4, lines 5-6, page 7 lines 27-28, page 9 lines 19-20 and 26-28, page 11 lines 13-14 and 27-28 and 31, page 12 lines 1 and 5-6 and 10-11 and 14-16, page 14 lines 6-7 and 20-21, page 15 lines1-2). The hook routine determines a data flow or process requested by the system call (page 4 lines 6-7, page 10 lines 12-24, page 11 lines 13-15 and 27-29, page 12 lines 1-3 and 5-8 and 10-12 and 14-17, page 13 lines 7-13 and 16-21 and 23-26, page 14 lines 7-9 and 21-23, page 15 lines 3-5) and another data flow or process for data related to that of the call (decision steps 603, 613 and 623 in Fig. 6, page 4 line 7, page 10 lines15-19, page 11 lines 30-31, page 12 lines 4-5 and 9-10 and 13-14 and 17-19 and 24-26, page 13 lines 7-13 and 16-21 and 23-26, page 14 lines 9-14 and 23-29, page 15 lines 5-10), and automatically generates a consolidated information flow diagram showing the data flow or process of the call and the other data flow or process (300 in Fig. 3, 330 in Fig. 4, 500 in Fig 5, steps 604, 605, 614, 615, 624, 625 in Fig. 6, page 4 lines 8-9, page 10 lines 12-24, page 11 lines 1-31, page 12 lines 1-28, page 13 lines 7-27, page 14 lines 1 and 14-15 and 27-29, page 15 lines 8-10). Then, the hook routine calls a routine to perform the data flow or process requested by the system call of step 608 (step 212 of Fig. 2B, step 628 of Fig. 6, page 4 lines 9-10, page 11 lines 26-27, page 12 lines 28-30, page 13 lines 27-30, page 14 lines 16-17 and 29-30, page 15 lines 10-11).

(V.B) Claim 2

Claim 2, which depends from claim 1 is directed to a user monitoring the information flow diagram and comparing the data flow process with a data flow or process expected by the user (page 15 lines 13-25).

(V.C) Claim 3

Claim 3, which depends from claim 1, is directed to a process according to its parent claim and wherein the information flow diagram 300, 330, 500 illustrates

locations of the data (Fig. 3 - file 301, memory 303, register 304, register 308, second file 309, pages 11, 12; Fig. 4 - file A 310, bytes 311 in memory 312, file B 313, memory 314, register 316, bytes 317 in memory 318, file C 319, memory 320 and socket 321, page 14; Fig. 5 – memory locations 502,504, 506, pages 14, 15) at stages of a processing activity.

(V.D) Claim 4

Claim 4, which depends from claim 1 is directed to a process according to its parent claim and wherein the system call (step 205 in Figs. 2A, 2B, page 9 lines 11-12) is selected from the set of: open file, copy file to memory, copy memory to register, mathematical functions, write to file, and network or communication functions (page 8 lines 25-29, page 10 lines 8-10).

(V.F) Claim 5

Claim 5, which depends from claim 1 is directed to a process according to its parent claims and wherein the system call is a software interrupt of an operating system (step 202 in Figs. 2A, 2B, page 8 lines 9-18, page 9 lines 8-10 and 17-19, page 10 lines 8-10).

(V.F) Claim 8

Claim 8 is directed to an apparatus (Fig. 1B, page 7 lines 19-24) for detecting malicious software within or attacking a computer system. The apparatus comprises a computer system 110 (Fig. 1A, page 7 line 19) with a display screen 120 (Fig. 1, page 7 line 19).

The computer system executes a program of instructions (Fig. 2B, Fig. 6, Fig. 7A, Fig. 7B, page 1 lines 7-9, page 3 lines 13-14, page 4 lines 1-18, page 7 lines 3-10). The program comprises executing a hook routine (step 211 in Fig.2B, 600, 610, 620 Fig. 6, page 4 lines 2-10, page 7 lines 26-31, page 9 lines 19-20, page 11 lines 13-31, page 12 lines 1-30, page 13 lines 7- 30, page 14 lines 7-31, page 15, lines 1-25) in response to a

system call (step 206 in Fig 2B, steps 601, 611, 621 in Fig. 6, page 4 lines 5-6, page 9 lines 17-20 and 26-28, page 11 lines 13-15 and 27-29 and 31, page 12 lines 1-3 and 5-8 and 10-12 and 14-17, page 14 lines 6-7 and 20-23, page 15 lines 1-5). The hook routine is located at the location identified by the system call (page 4, lines 5-6, page 7 lines 27-28, page 9 lines 19-20 and 26-28, page 11 lines 13-14 and 27-28 and 31, page 12 lines 1 and 5-6 and 10-11 and 14-16, page 14 lines 6-7 and 20-21, page 15 lines1-2). The hook routine determines a data flow or process requested by the system call (page 4 lines 6-7, page 10 lines 12-24, page 11 lines 13-15 and 27-29, page 12 lines 1-3 and 5-8 and 10-12 and 14-17, page 13 lines 7-13 and 16-21 and 23-26, page 14 lines 7-9 and 21-23, page 15 lines 3-5) and another data flow or process for data related to that of the call (decision steps 603, 613 and 623 in Fig. 6, page 4 line 7, page 10 lines15-19, page 11 lines 30-31, page 12 lines 4-5 and 9-10 and 13-14 and 17-19 and 24-26, page 13 lines 7-13 and 16-21 and 23-26, page 14 lines 9-14 and 23-29, page 15 lines 5-10), and automatically generates a consolidated information flow diagram showing the data flow or process of the call and the other data flow or process (300 in Fig. 3, 330 in Fig. 4, 500 in Fig 5, steps 604, 605, 614, 615, 624, 625 in Fig. 6, page 4 lines 8-9, page 10 lines 12-24, page 11 lines 1-31, page 12 lines 1-28, page 13 lines 7-27, page 14 lines 1 and 14-15 and 27-29, page 15 lines 8-10). Then, the hook routine calls a routine to perform the data flow or process requested by the system call of step 608 (step 212 of Fig. 2B, step 628 of Fig. 6, page 4 lines 9-10, page 11 lines 26-27, page 12 lines 28-30, page 13 lines 27-30, page 14 lines 16-17 and 29-30, page 15 lines 10-11).

(V.G) Claim 9

Claim 9, which depends from claim 8, is directed to a system according to its parent claims and wherein the information flow diagram 300, 330, 500 illustrates locations of the data (Fig. 3 - file 301, memory 303, register 304, register 308, second file 309, pages 11, 12; Fig. 4 - file A 310, bytes 311 in memory 312, file B 313, memory 314, register 316, bytes 317 in memory 318, file C 319, memory 320 and socket 321, page 14; Fig. 5 – memory locations 502,504, 506, pages 14, 15) at stages of a processing activity.

(V.H) Claim 10

Claim 10, which depends from claim 8, is directed to a system according to its parent claim and wherein the system call (step 205 in Figs. 2A, 2B, page 9 lines 11-12) is selected from the set of: open file, copy file to memory, copy memory to register, mathematical functions, write to file, and network or communication functions (page 8 lines 25-29, page 10 lines 8-10).

(V.I) Claim 11

Claim 11, which depends from claim 8, is directed to a system according to its parent claims and wherein the system call is a software interrupt of an operating system (step 202 in Figs. 2A, 2B, page 8 lines 9-18, page 9 lines 8-10 and 17-19, page 10 lines 8-10).

(V.J) Claim 14

Claim 14 is directed to a program product (page 15 lines 27-31) for detecting malicious software within or attacking a computer system (page 1 lines 7-9, page 3 lines 13-14, page 4 lines 1-18, page 7 lines 3-10). The program product executes a method that comprises executing a hook routine (step 211 in Fig.2B, 600, 610, 620 Fig. 6, page 4 lines 2-10, page 7 lines 26-31, page 9 lines 19-20, page 11 lines 13-31, page 12 lines 1-30, page 13 lines 7- 30, page 14 lines 7-31, page 15, lines 1-25) in response to a system call (step 206in Fig 2B, steps 601, 611, 621 in Fig. 6, page 4 lines 5-6, page 9 lines 17-20 and 26-28, page 11 lines 13-15 and 27-29 and 31, page 12 lines 1-3 and 5-8 and 10-12 and 14-17, page 14 lines 6-7 and 20-23, page 15 lines 1-5). The hook routine is located at the location identified by the system call (page 4, lines 5-6, page 7 lines 27-28, page 9 lines 19-20 and 26-28, page 11 lines 13-14 and 27-28 and 31, page 12 lines 1 and 5-6 and 10-11 and 14-16, page 14 lines 6-7 and 20-21, page 15 lines1-2). The hook routine determines a data flow or process requested by the system call (page 4 lines 6-7, page 10 lines 12-24, page 11 lines 13-15 and 27-29, page 12 lines 1-3 and 5-8 and 10-12 and 14-

17, page 13 lines 7-13 and 16-21 and 23-26, page 14 lines 7-9 and 21-23, page 15 lines 3-5) and another data flow or process for data related to that of the call (decision steps 603, 613 and 623 in Fig. 6, page 4 line 7, page 10 lines15-19, page 11 lines 30-31, page 12 lines 4-5 and 9-10 and 13-14 and 17-19 and 24-26, page 13 lines 7-13 and 16-21 and 23-26, page 14 lines 9-14 and 23-29, page 15 lines 5-10), and automatically generates a consolidated information flow diagram showing the data flow or process of the call and the other data flow or process (300 in Fig. 3, 330 in Fig. 4, 500 in Fig 5, steps 604, 605, 614, 615, 624, 625 in Fig. 6, page 4 lines 8-9, page 10 lines 12-24, page 11 lines 1-31, page 12 lines 1-28, page 13 lines 7-27, page 14 lines 1 and 14-15 and 27-29, page 15 lines 8-10). Then, the hook routine calls a routine to perform the data flow or process requested by the system call of step 608 (step 212 of Fig. 2B, step 628 of Fig. 6, page 4 lines 9-10, page 11 lines 26-27, page 12 lines 28-30, page 13 lines 27-30, page 14 lines 16-17 and 29-30, page 15 lines 10-11).

(V.K) Corresponding Structure for Means of Claim 8

The structure corresponding to means for executing a hook routine is the computer system 110 operating a program of instructions (step 211 in Fig. 2B, page 4 lines 1-18, page 9 lines 17-20) and equivalents.

The means for displaying the information flow diagram is the display screen 120 (Fig. 1, page 7 line 19) and equivalents.

(VI). Grounds of Rejection to be reviewed on appeal

Each of claims 1-15 is rejected under 35 U.S.C. 102 as being anticipated by U.S. Patent No. 6,823,460 to Hollander et al. (hereafter Hollander).

The questions for appeal are whether or not each of claims 1-15 is anticipated by Hollander under 35 U.S.C. 102.

(VII). Argument

(VII.A) Principles of Law Relating to Anticipation

The Examiner must make a prima facia case of anticipation. "A person shall be entitled to a patent unless. . . (b) the invention was patented or described in a printed publication in this or a foreign country . . . more than one year prior to the date of the application for patent in the United States." 35 U.S.C. 102. It is settled law that each element of a claim must be expressly or inherently described in a single prior art reference to find the claim anticipated by the reference. "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." Verdegaal Bros. v. Union Oil Co. of California, 814 F.3d 63, 631, 2USPQ2d 1051,1053 (Fed. Cir. 1987), cert. denied, 484 U.S. 827 (1987). Inherency, however, may not be established by probabilities or possibilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient." In re Robertson, 169 F.3d 743, 745, 49 USPQ2d 1949, 1951 (Fed. Cir. 1999)(citations and internal quotation marks omitted). The Examiner has failed to make a prima facia case of anticipation, because the claims on appeal include various elements that are not expressly or implicitly described in the reference cited (i.e., Grimm).

(VII.B) Rejection of Claims 1, 14 under 35 USC 102 over US Patent 6,823,460 (Hollander)

Applicants have contended that claims 1 and 14, as originally filed are allowable because they include features that are neither disclosed nor suggested by Hollander or any other references, either individually or in combination.

(VIII.B.1) ***"in response to a system call, executing a hook routine at a location of said system call"***

Claim 1 includes a first element "in response to a system call, executing a hook routine at a location of said system call," and claim 14 includes the similar element "program instructions responsive to a system call for executing a hook routine at a location of said system call."

As described in the present application, in an exemplary embodiment a system call is an operation which transfers control of a processor, such as by stopping the current processing in order to request a service provided by an interrupt handler. The interrupt handler in turn is a program code at a memory location identified by an interrupt vector table. In this example, the system call comprises an interrupt. By use of the interrupt vector table, the system executes the software routine located at the memory location designated (or pointed to) for the particular interrupt in the interrupt vector table. In claim 1, in response to a program making a system call during execution of the program, the processor is pointed to a memory location for the system call. The hook routine, as provided in claim 1, is a code segment located at the location pointed to by the system call, and is executed in response to the system call. This is significant because the programming level at which software interrupts work is the level at which many computer viruses work (see specification page 8, line 30 to page 9, line 1). Moreover, the present application clearly indicates that it is desirable to hook the lowest level calls where possible (see specification page 9, lines 4-5). In the present invention, the hook routine is a code segment separate from interrupt handler or API. The hook routine neither takes control of an API nor modifies an API. Rather control of the processor is switched to the hook routine during execution of an application by virtue of the hook routine's location at the address pointed to in an interrupt vector table.

Hollander does not disclose or suggest executing a hook routine in response to a system call. In fact, Hollander defines hooking (for the purpose of interpreting their specification) as obtaining control of a desired API (col. 8, lines 37-39). Moreover, Hollander describes a hooking and patching process in which a module takes control of an API and patches or modifies portions of it (col. 7 lines 51-61). An API interception module 32 then handles API calls (Col. 7 lines 56-58). Hollander does not execute a

hooking routine in response to a system call, but rather overwrites a section or sections of the API to control the code behavior (col. 8, lines 40-44). Applicants respectfully contend that overwriting the API does not disclose or suggest executing a hook routine at a location of the system call. Moreover, taking control of an API and patching it is not the same as executing a hook routine that is located at an address pointed to by a vector intercept table.

Hollander also fails to disclose or suggest that a hook routine is executed in response to a system call. The hook and patch process of Hollander is performed in response to loading an API intercept module 32 (Col. 7 lines 51-53) not in response to a system call.

The Examiner argues in error that this feature is disclosed by Hollander at col. 6, lines 7-11. The cited text provides that an API Interception Control Server monitors the host operating system for system calls through a system call interception component. Hollander does not suggest a hook routine at the location of the system call. Rather an intercept control server monitors the host operating system form system calls and takes actions based on the type of system call. The action taken is a modification or overwriting of the API, and not the execution of a hook routine. Monitoring for a system call is not the same as executing a hook routine in response to a system call. Nor is modification of an API the same as locating a hook routine at the address pointed to for an interrupt handler.

Applicants have contended that Hollander is silent regarding the operation of the interception component. The Examiner argues that the operation is disclosed at col. 7 line 65 to col. 8 line 30. In Hollander the interception component accesses a pre-defined API descriptor structure (which is a user-defined table) and allocates address space for API Flow Structure. However, API flow structure is not defined. Moreover, as described in Hollander at col. 8 line 58 to col. 9 line 54 portions of the API are overwritten with redirection code. Hollander does not specify where the redirection code comés from, how it is created, or how it operates. As best understood the table contains a listing of

functions to be performed before or after the API operation, but the actual redirection comes from modification of the API. The Examiner may not assume a method to establish anticipation. It is not sufficient that a certain thing may result from a given set of circumstances (see In Re Robertson, 169 F.3d 743, 745, 49 USPQ2d 1949, 1951 (Fed. Cir. 1999)(citations and internal quotes omitted).

The Examiner has failed to make a prima facia case that the element "in response to a system call, executing a hook routine at a location of said system call," is anticipated by Hollander. Nor is this feature obvious in view of Hollander as Hollander neither discloses nor suggests executing a hook routine in response to a system call.

***(VII.B.2)* "determine a dataflow or process requested by said call"**

Claims 1 and 14 include a second element "determine a dataflow or process requested by said call."

In the present application, in response to a system call being made, the hooking routine executes to monitor and display the operation of the computer system (see specification page 10, lines 1-8). The hooking routines generate an icon or other graphical representation of the current operation to be performed by the original routine (i.e., the routine called by the system call). Thus, in claims 1 and 14, the hooking routine determines the actual function to be performed by the system call (e.g., transferring data, mathematical function, deleting data, copying data, etc.). As pointed out previously, it is important to monitor the system call functions, as this is where many malicious software programs operate.

The Examiner argues in error that Hollander discloses this feature at col. 6, lines 7-11, discussed above. Hollander does not disclose or suggest determining a data flow or process requested by a system call, much less a hook routine that determines a data flow or process requested by the call.

The Examiner has failed to make a prima facia case of anticipation of the second element, "determine a data flow or process requested by said call." Nor is this feature obvious in view of Hollander as Hollander neither discloses nor suggests a hook routine determining a data flow or a process flow requested by a system call.

(VII.B.3) ***"determine another data flow or process for data related to that of said call"***

Claims 1 and 14 include a third element, "determine another data flow or process for data related to that of said call," which Applicant contend are neither disclosed nor suggested by Hollander or any other reference.

The present invention provides for associating a current system call function with another system call function by matching file names or memory locations, for example. This step is important to creating a meaningful process flow to track a data flow or process, as operations that in isolation may not appear malicious, but viewed together, show a malicious pattern. Malicious software may manipulate data, for example, using a series of steps which individually do not appear malicious. By following the data or process flow the danger, however, becomes apparent.

Hollander does not disclose or suggest stringing together related system call operations to track data flow or process flows. The Examiner apparently argues that this feature is provided by Hollander at col. 6, lines 12-20. Applicants respectfully disagree. The cited text merely recites that an API interception control server monitors a host operating system for system calls and takes an action according to the type of system call. Hollander is silent as to how the monitoring is performed. Moreover, Hollander does not disclose or suggest any determination of a data or process flow or determination of related data or process flow, much less stringing together related data or process flows.

The Examiner has failed to make a prima facia case of anticipation of the third element, "determine another data flow or process for data related to that of said call."

Nor is this feature obvious in view of Hollander as Hollander neither discloses nor suggests a hooking routine determining a related data flow to that of a system call.

(VII.B.4) "*automatically generate a consolidated information flow diagram showing said data flow or process of said call and said other data flow or process*"

Claims 1 and 14 include a fourth element, "automatically generate a consolidated information flow diagram showing said data flow or process of said call and said other data flow or process," that is neither disclosed nor suggested by Hollander or any other reference.

This element is directed to the hooking program combining the data flow of a current system call with the data flow of related system calls and presenting this data flow information in a specific useful form, namely a consolidate information flow diagram. A diagram is a graphical representation, as shown in Figs. 3, 4, and 5 and described in the specification at pages 11-15.

Hollander does not disclose or suggest generating a consolidated information flow diagram. The Examiner argues that Hollander discloses this feature in Fig. 7, step 156 contending that an API flow table is analogous to a consolidated information flow diagram. Applicants respectfully disagree. Hollander does not disclose or suggest a data or process flow diagram that consolidates data or process flow over multiple system calls. Moreover, Applicants respectfully content that the API Flow Structure Table is not defined in Hollander, and that Hollander is unclear regarding what the API Flow Structure Table is and how it functions.

The Examiner has failed to make a prima facia case of anticipation with respect to the fourth element, "automatically generate a consolidated information flow diagram showing said data flow or process of said call and said other data flow or process." Nor is this feature obvious in view of Hollander as Hollander neither discloses nor suggests

automatically generating a consolidated information flow diagram showing the data flow or process flow requested by a system call and a related data flow or process flow.

(VII.C) Rejection of Claim 8 under 35 USC 102 over US Patent 6,823,460 (Hollander)

Applicants have contended that claim 8, as originally filed is allowable because it includes features that are neither disclosed nor suggested by Hollander or any other references cited in the First Office Action, either individually or in combination.

(VII.C.1) ***elements previously discussed under claim 1***

"means responsive to a system call, for executing a hook routine at a location of said system call"

This element is directed to a computer operating a program of instruction substantially similar to the first element argued under claims 1 and 14. These arguments will not be repeated here.

"determine a dataflow or process requested by said call"

This element is directed to a computer operating a program of instruction substantially similar to the second element argued under claims 1 and 14. These arguments will not be repeated here.

"determine another data flow or process for data related to that of said call"

This element is directed to a computer operating a program of instruction substantially similar to the third element argued under claims 1 and 14. These arguments will not be repeated here.

"automatically generate a consolidated information flow diagram showing said data flow or process of said call and said other data flow or process"

This element is directed to a computer operating a program of instruction substantially similar to the fourth element argued under claims 1 and 14. These arguments will not be repeated here.

(VII.C.2) *"means for displaying said information flow diagram"*

Claim 8 includes the element "means for displaying said information flow diagram." The structure associated with this element is the display screen 120. Thus, the information flow diagram representing the information flow determined by computer 110 is displayed on display screen 120. Hollander does not describe either an information flow diagram or a display screen. Accordingly, the Examiner has failed to make a prima facia case of anticipation. Nor is this feature obvious in view of Hollander as Hollander neither discloses nor suggests a means for displaying an information flow diagram.

(VII.D) Rejection of Claim 2 under 35 USC 102 over Patent 6,823,460 (Hollander)

Applicants have contended that claim 2, as originally filed is allowable independently of its parent claim 1, because it includes features that are neither disclosed nor suggested by Hollander or any other references cited in the First Office Action, either individually or in combination.

(VII.D.1) "a user monitors said information flow diagram"

Claim 2 includes the element "a user monitors said information flow diagram." Hollander does not disclose or suggest a user monitoring an information flow diagram. A user monitoring the information flow diagram is significant because it allows the user to detect suspicious activity in real time and take remedial measures (page 15 lines 13-25).

The Examiner argues in error that this feature is provided by Hollander at col. 2, lines 45-52. This text does not provide a user or a flow diagram, much less a user monitoring an information flow diagram.

The Examiner has failed to make a prima facia case of anticipation with respect to the fourth element, "a user monitors said information flow diagram." Nor is this feature obvious in view of Hollander as Hollander neither discloses nor suggests a user monitoring an information flow diagram.

(VII.D) Rejection of Claims 3 and 9 under 35 USC 102 over Patent 6,823,460 (Hollander)

Applicants have contended that claims 3 and 9, as originally filed are allowable independently of their parent claims 1 and 8, respectively, because they include a feature that is neither disclosed nor suggested by Hollander or any other references cited in by the Examiner, either individually or in combination.

(VII.D.1) ***"said information flow diagram illustrates locations of said data at stages of a processing activity"***

Claims 3 and 9 include the element "said information flow diagram illustrates locations of said data at stages of a processing activity." The present invention provides an embodiment in which the flow of data is captured and the location of the data at stages of processing is illustrated. This allows a user to quickly see where data is being transferred to, and therefore, whether data is being manipulated in an undesirable way. Hollander does not disclose or suggest illustrating locations of data during processing.

The Examiner argues in error that this feature is disclosed by Hollander in Fig. 3 , elements 154-165. Applicants respectfully disagree. Fig. 3 does not include elements 154-165. Moreover, Fig. 3 is directed to injecting an API Interception Module into

address space of an active process (see col. 6, lines 40 et. Seq.) and not to a flow diagram that illustrates the location of specific data during different phases of processing.

The Examiner has failed to make a prima facia case of anticipation with respect to the fourth element, "said information flow diagram illustrates locations of said data at stages of a processing activity." Nor is this feature obvious in view of Hollander as Hollander neither discloses nor suggests an information flow diagram illustrating locations of data at stages of a processing activity.

Respectfully Submitted,

Steven E. Bach
Attorney for Applicants
Reg. No. 46,530

(VIII) Claims Appendix

Listing of Claims:

1. (original) A method for detecting malicious software within or attacking a computer system, said method comprising the steps of:

In response to a system call, executing a hook routine at a location of said system call to (a) determine a data flow or process requested by said call, (b) determine another data flow or process for data related to that of said call, (c) automatically generate a consolidated information flow diagram showing said data flow or process of said call and said other data flow or process, and after steps (a-c), (d) call a routine to perform said data flow or process requested by said call.

2. (original) A method as set forth in claim 1, wherein a user monitors said information flow diagram and compares the data flow process of steps (a) and (b) with a data flow or process expected by said user.

3. (original) A method as set forth in claim 1, wherein said information flow diagram illustrates locations of said data at stages of a processing activity.

4. (original) A method as set forth in claim 1, wherein said system call is selected from the set of: open file, copy file to memory, copy memory to register, mathematical functions, write to file, and network or communication functions.

5. (original) A method as set forth in claim 1, wherein said system call is a software interrupt of an operating system.

6. (original) A method as set forth in claim 1, wherein said system call causes a processor to stop its current activity and execute said hook routine.

7. (original) A method as set forth in claim 1 wherein said system call is made by malicious software.

8. (original) A system for detecting malicious software in a computer system, said system comprising:

means, responsive to a system call, for executing a hook routine at a location of said system call to (a) determine a data flow or process requested by said call, (b) determine another data flow or process for data related to that of said call, (c) automatically generate a consolidated information flow diagram showing said data flow or process of said call and said other data flow or process, and after steps (a-c), (d) call a routine to perform said data flow or process requested by said call; and

means for displaying said information flow diagram.

9. (original) A system as set forth in claim 8, wherein said information flow diagram illustrates locations of said data at stages of a processing activity.

10. (original) A system as set forth in claim 8, wherein said system call is selected from the set of: open file, copy file to memory, copy memory to register, mathematical functions, write to file, and network or communication functions.

11. (original) A system as set forth in claim 8, wherein said system call is a software interrupt of an operating system.

12. (original) A system as set forth in claim 8, wherein said system call causes a processor to stop its current activity and execute said hook routine.

13. (original) A system as set forth in claim 8 wherein said system call is made by malicious software.

14. (original) A computer program product for detecting malicious software in a computer system, said computer program product comprising:

a computer readable medium;

program instructions, responsive to a system call, for executing a hook routine at a location of said system call to (a) determine a data flow or process requested by said call, (b) determine another data flow or process for data related to that of said call, (c) automatically generate a consolidated information flow diagram showing said data flow or process of said call and said other data flow or process, and after steps (a-c), (d) call a routine to perform said data flow or process requested by said call; and wherein said program instructions are recorded on said medium.

15. (previously presented) A method as set forth in claim 1, wherein said hook routine is at a location pointed to by said system call.

(IX). Evidence appendix

No extrinsic evidence is presented.

(X). Related proceedings appendix

There are no related proceedings.